METHODS AND APPARATUS FOR SELF-DESCRIBING DEVICES

Field of the Invention

The present invention generally relates to devices, such as hardware components, for being integrated with a computer system and arrangements and methods for integrating such devices with a computer system.

Background of the Invention

In the context of many known operating systems, it is possible to integrate different devices, such as hardware components, with the operating system through device drivers. Device drivers are low-level software programs that translate abstract device commands, as specified in the operating system's device interface, into actual commands issued to the hardware device.

One problem commonly encountered is that device drivers must be updated in order to eliminate bugs, to improve the performance of the integrated device or to enhance the functionality of the device. The updating of device drivers is typically accomplished by obtaining a new device driver from the device manufacturer, e.g., on a storage device such as a diskette or CD-ROM or via the downloading of one or more device driver files from the manufacturer's web site.

Sub A1

In the realm of software, as well, certain arrangements for updating are known. Examples include Intuit's *TurboTax*, where the software periodically queries the user whether to check the Intuit web page for updated versions of the software and/or newer versions of the tax forms. With Norton's *AntiVirus*, the software periodically checks whether updated virus signature files are available on the company web site. With the IBM *Global Network Dialer*, when there is a connection to the internet, the software determines whether a newer version of the software and/or a newer version of the phone number list is available and asks the user whether to update.

None of the software programs just described, however, implement the concept of storing the necessary software components within a physical device itself. Furthermore, all of the programs in question rely on proprietary solutions to implement the function of automatic updates, instead of using a common protocol to test and update existing versions of the software.

To a degree, dynamic reconfiguration is available today (e.g., plug 'n' play of PC-cards) and will likely become more widespread over the next few years. However, when integrating new devices into a computer system one or more of the following problems often occurs:

- The operating system does not know the type of device and does not have the proper device driver file or files.

- Although the operating system knows about the device, the device driver is not available on the system and must be loaded from the operating system's install medium (e.g., tape, diskette, CD-ROM)

- The operating system knows the device but a back-level (old) version of the device driver is available within the system.

In view of the foregoing, a need has been recognized in connection with providing arrangements and methods for more efficiently configuring a computer system with new hardware components.

Summary of the Invention

The present invention, in accordance with at least one presently preferred embodiment, overcomes the disadvantages and shortcomings of previous efforts by storing the device driver on the device itself from where it can be loaded by the operating system.

Preferably, the information stored on the device will have at least the following three components:

- A description of the device through a unique ID and some textual, human-readable description.

- The device driver file(s) with version numbers.

- Information on how to obtain newer versions of the device driver, e.g., a URL.

Preferably, the aforementioned information will be exchanged over a common protocol supported by the operating system and all such devices.

For a better understanding of the present invention, together with other and further features and advantages thereof, reference is made to the following description, taken in conjunction with the accompanying drawings, and the scope of the invention will be pointed out in the appended claims.

Brief Description of the Drawings

Figure 1 schematically illustrates a computer system with attached device.

Figure 2 schematically illustrates the internal structure of a device.

Figure 3 illustrates a flow diagram that sets forth a protocol for integrating a device with an operating system.

Figure 4 schematically illustrates the integration of a device with into typical device driver stacks.

Description of the Preferred Embodiments

Figure 1 shows a schematic high-level view of a computer system 101 with an attached device 102. The device 102 is connected to the computer system 101 via an interface 101. For example, the device 102 can be a graphics adapter or a disk drive and the interface can be a PCI bus or an IDE connection. Essentially, device 102 could be any component originally separate from computer system 100 that is to be subsequently integrated with the computer system 100 and for which, conventionally, a rather involved installation procedure may be required. Thus, for instance, device 102 could also be a modem, a printer, a sound card, a video card, a graphics card, a IDE or SCSI adapter, a network interface card, a network controller, a CD ROM drive, or a tape drive.

In Figure 2, the internal structure of the device 102 is schematically shown. Preferably, in addition to the actual functioning components of the device, e.g., device logic, depicted by block 203, there is also a device description subsystem 200. This subsystem 200 preferably includes two main parts: a non-volatile, read-write memory 201 and interface logic 202. The interface logic 202 interprets commands received over the

computer-device interface 101 and controls the non-volatile memory 201, which contains the locally stored device driver.

The commands understood by the interface logic 202 preferably support the following functions:

I. *Device Identification*: This function returns a unique identifier and a textual device name. The identifier allows the computer system and its operating system to positively identify the device. The name can be used by the operating system to present to the user the device(s) attached to the computer system.

II. *Device Driver Versions*: This function returns version information about the device driver stored on the device. Versions are defined e.g., by a triplet of numbers that indicate major version, minor version and release number.

III. *Read Device Drivers*: This function allows the operating system to obtain a copy of the device driver for installation.

IV. *Get Link to Device Driver Data*: This function provides a reference to a network location, e.g., a URL, where the most recent version of the device driver can be found.

V. Update Device Drivers: This function allows updating the device driver information stored on the device with a more recent version.

VI. Update Link: This function allows to update the location of where to find the latest version of a given device driver. (Preferably, the device manufacturer will maintain such locations with care in order not to send queries to non-existing sites.)

Figure 3 is a flow diagram that illustrates how the operating system interacts with the device in accordance with an embodiment of the present invention. If the computer system detects a new device (step 301) through essentially any suitable known method, it first determines the device identification (302). Based on the device identification, the operating system can decide whether this device is already known (303). If not, it retrieves the link from the device (304); otherwise that link is already known. Then, a test (305) is performed to determine whether the link is accessible; e.g., if the computer system is not attached to a network a URL cannot be resolved. If the link is not accessible, then the existing local device driver ("old" device driver) will be used. If the link is accessible, however, the available (local) driver version is tested against the remote version to find out whether a more recent one is available (307). The newer one of the remote and local version is then installed by the operating system [OS] (309, 313) and if necessary downloaded into the device (310).

Figure 4 shows how device driver stacks, as they are presently used in many operating systems (e.g., Microsoft Windows), can integrate a device description arrangement in accordance with at least one embodiment of the present invention. The physical device drivers (403), e.g., a PCI (peripheral component interconnect) bus driver or a SCSI (small computer system interface) driver is accessed by two higher-level drivers. The device-specific driver 402 is the standard device driver that addresses the actual function of the device. The generic device driver 401 in conjunction with the operating system 400 implements the algorithm of Figure 3 described above.

Several practical scenarios can help illustrate the advantages that may be enjoyed in connection with a device describing arrangement provided and utilized in accordance with at least one embodiment of the present invention. For instance, if a device is connected to a stand-alone (i.e., not network connected) computer system, then the device driver can be loaded directly and automatically from the device without the need to find the OS installation CD.

If a device is connected to a network-connected computer, then the most recent device driver can be retrieved from the location stored on the device. If the latest device driver is already stored on the device, no lengthy download over the network is required as the device driver is retrieved directly from the device.

As another advantage, the device driver on a device can be updated with a newer driver from the network on a network-connected computer or from other media like a CD-ROM or diskette.

A device description arrangement, in accordance with at least one alternative embodiment of the present invention, can be used to configure any network devices or resources and install the necessary drivers. Such network devices or resources may include printers, fax machines, and scanners. In this case, the necessary drivers are stored inside the device or resource. Upon request or automatically, the computer system enumerates all the network devices or resources available locally and within a network neighborhood (e.g., on the same subnet or ring). All such network devices or resources would thus be automatically installed and made available to the user.

As a variation on the process described above and shown in Figure 3, a computer system may periodically check the currently installed driver against the version available on the remote site. If a newer driver is detected, it is downloaded, installed and updated on the device. The predetermined time intervals at which such periodic checking may take place can be chosen depending on the volatility associated with the driver in question. For example, it is conceivable to check for new versions of graphics and network card drivers at significantly frequent intervals, in view of the fact that these are typically

updated quite often, while versions of printer drivers, on the other hand, might not need to be checked upon quite so frequently since these do not tend to be updated very often.

A further variation on the basic algorithm described above and shown in Figure 3 resides in querying the user as to whether to download a new driver in cases of a low-bandwidth connection (e.g., phone line). The user may then decide to defer the download to a time when a better connection becomes available.

The process described and illustrated heretofore with respect to Figure 3 can be applicable both to cold-plugging (before power-on) and to hot-plugging (during operation) of new devices.

At least one presently preferred embodiment of the present invention broadly contemplates the installation of drivers over networks. Particularly, instead of storing the actual driver with the device, the device might only store a reference to a network location where to find the driver. For instance, for the purpose of facilitating integration with an operating system, a printer may have nothing more than an internet address that holds different printer drivers. The advantage of this setup is that it eliminates the need for checking the device driver version stored on the device and then conditionally updating it from the web. Instead, the most recent version would always be retrieved from the web. In this connection, less storage would be required on the device, the possibility would

arise of setting up a website with company-specific drivers, and the maintenance of devices would be easier (since there is only one location holding drivers).

It is to be understood that the present invention, in accordance with at least one presently preferred embodiment, includes an arrangement, inherent to a device, for facilitating the integration of the device with a computer operating system. The arrangement may be implemented in conjunction with at least one general-purpose computer running suitable software programs. It may also be implemented on at least one Integrated Circuit or part of at least one Integrated Circuit. Thus, it is to be understood that the invention may be implemented in hardware, software, or a combination of both.

If not otherwise stated herein , it is to be assumed that all patents, patent applications, patent publications and other publications mentioned and cited herein are hereby fully incorporated by reference herein as if set forth in their entirety herein.

Although illustrative embodiments of the present invention have been described herein with reference to the accompanying drawings, it is to be understood that the invention is not limited to those precise embodiments, and that various other changes and modifications may be affected therein by one skilled in the art without departing from the scope or spirit of the invention.